VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

REKONSTRUKCE ROZTŘÍŠTĚNÉHO OBJEKTU Z ÚLOMKŮ

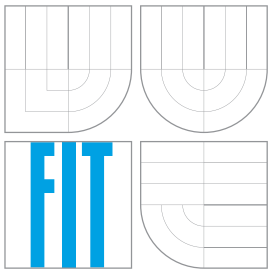
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

PETRA BAČÍKOVÁ

BRNO 2008

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

REKONSTRUKCE ROZTŘÍŠTĚNÉHO OBJEKTU Z ÚLOMKŮ

RECONSTRUCTION OF FRAGMENTED OBJECTS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

PETRA BAČÍKOVÁ

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. MICHAL ŠPANĚL

BRNO 2008

Abstrakt

Práce se zabývá rekonstrukcí roztříštěných objektů z úlomků. Obsahuje stručný přehled již realizovaných projektů s tematikou rekonstrukce. Dále popisuje základní metody zpracování obrazu, práci s konturami a korelaci. Jsou zde navrženy algoritmy pro zpracování kontur, detekci rohů a porovnávání hran.

Klíčová slova

Rekonstrukce roztříštěného obejktu z úlomků, zpracování obrazu, kontury, korelace, detekce rohů v kontuře, porovnávání hran, archeologie, C++, knihovna OpenCV, Python.

Abstract

Thesis deals with reconstruction of fragmented objects. It contains a sentenious view of already realized projects with reconstruction subject. In the folowing section, the thesis describes essential methods of image processing, work with contours and corelation. There are proposed algorithms of image processing, corners detection and edges matching.

Keywords

Reconstruction of fragmented objects, image processing, contours, corelation, corners detection, edges matching archeology, C++, OpenCV library, Python.

Citace

Petra Bačíková: Rekonstrukce roztříštěného objektu z úlomků, bakalářská práce, Brno, CZ, FIT VUT v Brně, 2008

Rekonstrukce roztříštěného objektu z úlomků

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracovala samostatně pod vedením Ing. Michala Španěla. Uvedla jsem všechny literární prameny a publikace, ze kterých jsem čerpala.

.....
Petra Bačíková
13. května 2008

Poděkování

Zde bych ráda poděkovala mému vedoucímu Ing. Michalu Španělovi za odborné vedení, poskytnuté konzultace a za vstřícnost při výběru zadání. Dále Mgr. Lucii Pokorné za poskytnutá testovací data a vědomosti z oblasti rekonstrukce v archeologii.

Obsah

Kapitola 1

Úvod

Práce popisuje metody a postupy používané k rekonstrukci roztříštěného objektu z úlomků. Hlavním cílem práce je vytvoření prostředku napomáhajícímu automatizovat práci archeologů. Tato činnost je silně závislá na správném zpracování testovacích dat a dále na zpětné vazbě od uživatelů prostředku. Výstupem je seznam dvojic objektů patřících s určitou mírou pravděpodobnosti k sobě.

Kapitola 2 má za úkol uvést čtenáře do problematiky rekonstrukce roztříštěných objektů v archeologii. Zde je třeba podotknout, že rekonstrukci je možno provádět ve dvou nebo trojrozměrném prostoru (2D – two dimensional, 3D – three dimensional). Tato práce podrobněji studuje pouze dvoudimenzionální prostor.

O základních pojmech při zpracování obrazu pojednává kapitola 3. Jedná se o redukci barevného prostoru. Jsou zde objasněny metody vyhledání kontur objektu v obraze, segmentace kontury a korelace.

4. kapitola se věnuje návrhu zpracování testovacích dat a výsledného programu. Popisuje jednotlivé etapy vývoje a úskalí, která během této doby nastala. Jsou zde uvedeny definice používaných pojmů a algotitmy používané při implementaci.

Na tuto kapitolu úzce navazuje kapitola 5, která popisuje výběr implementačních prostředků a samotnou implementaci programu včetně návodu na použití.

Testování programu je diskutováno v kapitole 6. Byly použity různé testovací objekty: uměle generované střepy v grafickém programu, exportované do rastrového obrázku, nafocené střepy z rozbitých talířů a hliněných nádob z archeologické lokace Pálava. Také je zde uvedeno, jak nejlépe testovací objekty vyfotografovat.

Závěr zhodnocuje výsledky práce, je diskutován budoucí vývoj projektu a jeho použití v praxi.

Kapitola 2

Rekonstrukce v archeologii

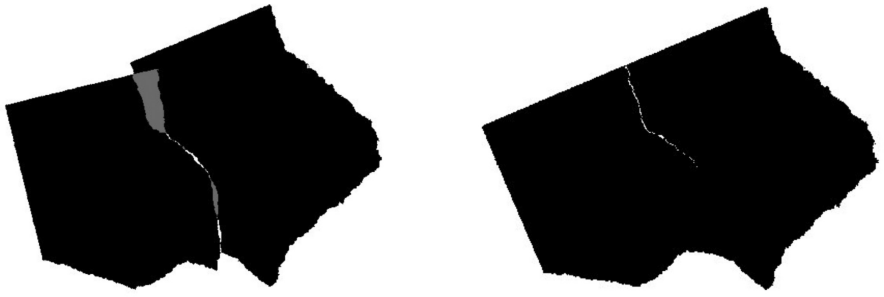
Pojem *rekonstrukce v archeologii* je možné chápat různě. Tato kapitola se zabývá rekonstrukcemi roztříštěného či jinak fragmentovaného objektu. Jsou zde popsány možnosti rekonstrukce podle tvarů objektu ve 2D nebo 3D prostoru, rekonstrukce podle barev a složení objektu.

2.1 Rekonstrukce podle tvarů

Dvoudimenzionální prostor

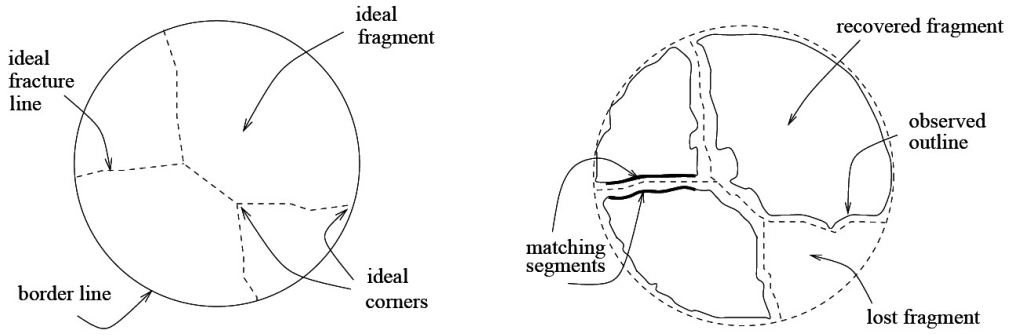
O zcela nových algoritmech pro rekonstrukci rozdělených 2D objektů na části se zabývá [7]. Je zde řešena efektivita metody globální rekonstrukce. Techniky jsou založeny na přesných mezifragmentových mezerách a detailním výpočtu překrytí fragmentů. Vyhledaná dvojce a její přesné sesazení k sobě jsou vidět v obrázku 2.1.

Další možnost [1] je rekonstrukce dokumentu, který je rozstříhaný nebo roztrhaný. Navrhovaná metoda nejprve postupně aplikuje polygonální aproximaci, aby byla odstraněna složitost hran, a poté vyhledává možné rysy polygonu, aby mohla být provedena lokální



a) b)

Obrázek 2.1: Vysokopreciznírekonstrukce (zdroj [7]), sesazení střepů ve dvou krocích: a) První přiřazení, b) Přesné sesazení



a) b)

Obrázek 2.2: Porovnání teoretických a skutečných střepů (zdroj [2]): a) Ideální fragmentace, b) Reálná fragmentace

rekonstrukce. Touto cestou se značně zredukuje celková složitost.

Účinný algoritmus pro rekonstrukci jednoho, nebo více neznámých objektů, které byly rozbity nebo rozlámány na velké množství nepravidelných fragmentů, je uveden ve zdroji [2]. Algoritmus sekvenčně porovnává hrany objektu v několika po sobě jdoucích krocích, přičemž v každém kroku se zvyšuje kvalita stupně rozlišení. Tím se zpočátku eliminují fragmenty, které určitě nepatří k sobě. Uvažuje se i počet fragmentů, který ovlivňuje počet průchodů.

Třídimenzionální prostor

O rekonstrukci v oblasti 3D prostoru pojednává [9]. Testovací data jsou získávána pomocí počítačové tomografie (CT – Computed Tomography). Tato radiologická metoda umožňuje zobrazení pomocí rentgenového záření. Ze získaných dat se pomocí matematického programu získávají segmenty, které se automaticky skládají. V této práci je dosaženo velice dobrých výsledků rekonstrukce roztříštěných kostí.

Jiný zdroj [3] uvádí rekonstrukci obličeje podle tvaru lebky. Porovnává testovací data získaná pomocí již zmíněné počítačové tomografie a data získaná pomocí barevného laserového skeneru (CLS – Color Laser Scanner), který zachycuje povrch objektu pomocí laserového paprsku a postupného otáčení objektu. Dle výsledků rekonstrukce oběma způsoby je patrné, že technologie CT je přesnější.

Počítačovým tomografem byla skenována i mumie faraona Tutanchamona. Bylo vytvořeno 1700 snímků od hlavy k patě [10]. Byla prováděna rekonstrukce obličeje podle vytvořených snímků. Na rekonstrukci se podílely tři na sobě nezávislé týmy (francouzský, americký a nakonec i egyptský), nicméně dosáhli poměrně podobných výsledků různými metodami. Přičemž pouze francouzský tým věděl, *čí* data zpracovává. Na výsledky rekonstrukce se můžete podívat na obrázku 2.3.

a) b) c)

Obrázek 2.3: Výsledky týmů z rekonstrukce obličeje faraona Tutanchamona, zdroj: Nevyšší rada pro památky, Egypt; a) egyptský, b) americký, c) francouzský

2.2 Rekonstrukce podle barev a složení

Tyto rekonstrukce spolu úzce souvisejí. Jsou využívány jako pomocné metody při rekonstrukcích podle tvaru objektu.

Barva nebo vzor na úlomcích mohou značně urychlit celý proces rekonstrukce. Příkladem z praxe je zpracování střepů hliněných nádob pocházejících z archeologické lokace Pálava. Nejprve jsou střepy roztříděny podle barvy, na kterou má vliv složení hlíny a míra vypálení nádoby, poté se třídí podle vzorů (ornamentů) a profilu střepu. Až nakonec přichází na řadu rekonstrukce, která se aplikuje na roztříděné množiny střepů. Tím se celý proces značně zjednoduší a urychlí.

Rozpoznání podle složení je využíváno v případech, kde jsou objekty stejné velikosti a tvaru. Zde hraje roli hlavně geologické složení u rekonstrukcí soch, chrámů apod. Naopak biologické složení hraje roli u rekonstrukce kostí.

Kapitola 3

Zpracování obrazu

Tato kapitola je teoretickým úvodem do problematiky zpracování obrazu. Postupně jsou v ní vysvětleny principy používané v praxi.

3.1 Obraz a digitalizace

Dříve než je objekt zpracován, musí se nějakým způsobem jeho obraz převést do digitální podoby, nejčastěji pomocí digitálního fotoaparátu nebo skeneru. Digitalizace je převod analogového signálu na diskrétní. Získaný obraz je uložen do počítače.

Obraz je reprezentován maticí hodnot, přičemž jeden prvek matice odpovídá jedné zobrazovací jednotce na monitoru – *pixelu*. Hodnota pixelu je různá pro barevný a šedotónový obraz. Pro barevný obraz obsahuje nejčastěji tři hodnoty spektrálních složek RGB (R – *red*, červená, G – *green*, zelená, B – *blue*, modrá). V šedotónovém obraze odpovídá hodnota pixelu pouze jedné hodnotě, a to intenzitě jasu. Při rozpoznávání kontur nás zajímá černobílý obraz, a proto je vstupní barevný obraz ihned zbaven barevných složek.

3.2 Převod barevného obrazu do šedotónového

Barevný obraz je ve standardních případech reprezentován jako 24 bit model. Šedotónový obraz je 8 bit model. Vzhledem k nedokonalosti lidského oka, které je schopné rozpoznat přibližně 50 odstínů šedé, je tento rozsah postačující [5].

Při převodu barevného obrazu na šedotónový je důležitý převodní vztah, který určuje, jak velký podíl budou mít jednotlivé barevné složky na intenzitu šedé barvy.

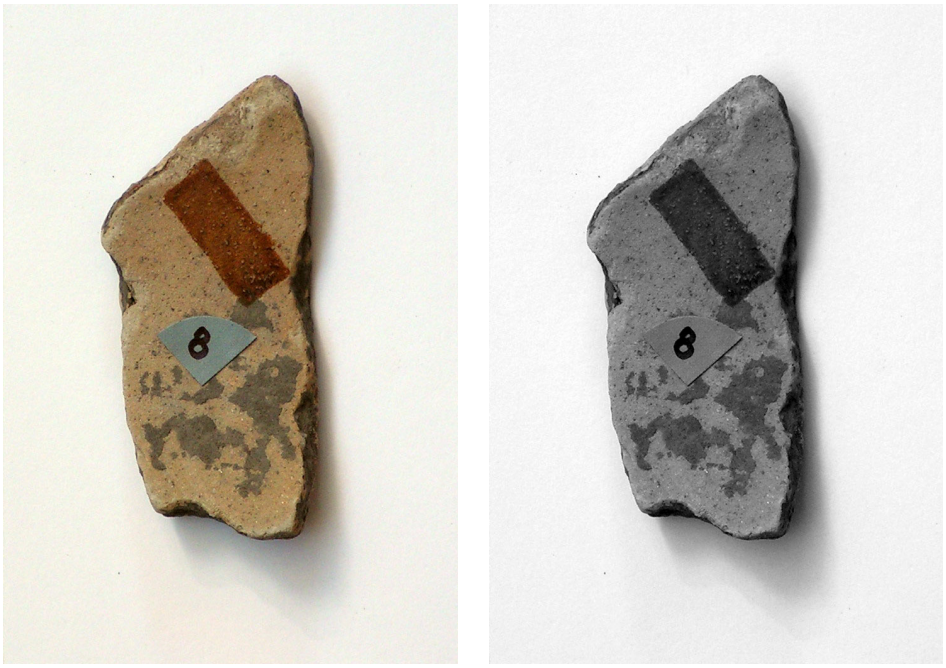
$$I = 0.229 * red + 0.587 * green + 0.114 * blue$$

kde I je intenzita šedi a red, $green$, $blue$ jsou spektrální složky barevného obrazu. Tento výpočet se při transformaci provádí pro každý pixel obrazu.

Algoritmus 3.1: *Převod barevného obrazu na šedotónový.*

- $G(x, y)$ je vstupní obraz, $G(x, y)$ je výstupní šedotónový obraz.
- Pro každý pixel barevného obrazu $C(x, y)$:

$$G(x, y) = 0.229 * C(x, y).red + 0.587 * C(x, y).green + 0.114 * C(x, y).blue$$



a) b)

Obrázek 3.1: a) Barevný obrázek (24b), b) Obrázek ve stupních šedi (8b)

3.3 Převod šedotónového obrazu na černobílý

Černobílý neboli *monochromatický* obraz obsahuje pouze dvě barvy – černou a bílou. Také se někdy takovému obrazu říká binární, protože pixel nabývá pouze dvou hodnot. Převod šedotónového obrazu na černobílý je možné provést několika způsoby popsanými níže.

Prvním způsobem je *dithering* – rozptylování. Tato metoda je založena na prostém nahrazení původních pixelů obrazu, za vhodnou distribuci černých a bílých bodů, přičemž je nutné maximálně zachovat vizuální podobu.

Druhý způsob převodu je *halftoning*, tzv. polotónování. Jedná se o způsob převodu, kde se původní pixel nahrazuje rozsahem hodnot vhodných distribucí černých a bílých bodů. Tím dochází k zvětšení rozlišení obrázku.

Pro účely práce byl vybrán první způsob převodu.

Thresholding

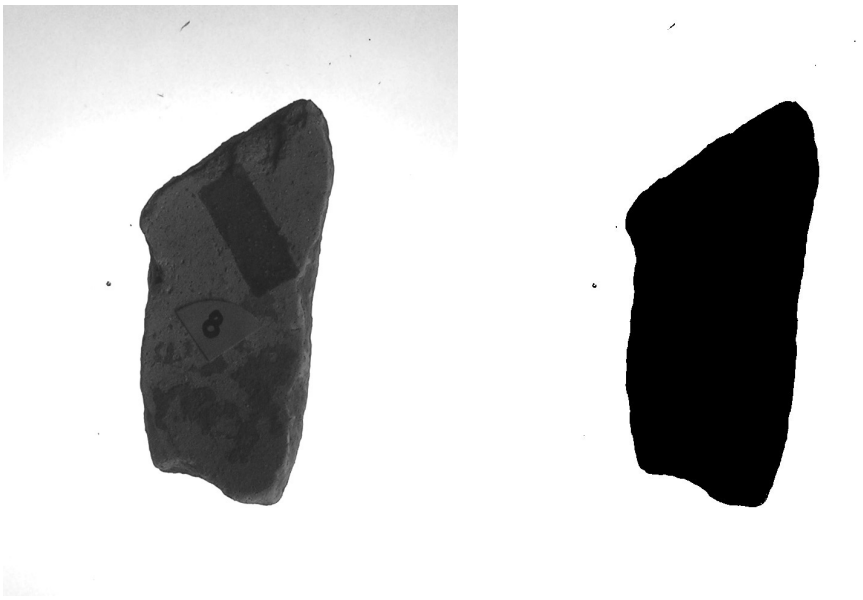
Thresholding, neboli prahování, je nejjednodušší metodou založenou na *ditheringu*. Je založena na porovnávání všech pixelů s prahem (*threshold*). Všechny pixely, které mají hodnotu intenzity pod daným prahem budou černé, ostatní budou bílé. Metoda je jednoduchá a rychlá.

Threshold je vhodné zvolit tak, aby nedošlo ke zkreslení vizuální podoby obrazu. Možnou metodou je počítání aritmetického průměru intenzity všech pixelů v obraze, díky čemuž nedojde k špatnému převedení jasnějších nebo tmavších obrazů.

Algoritmus 3.2: *Převod šedotónového obrazu na černobílý.*

- $G(x, y)$ je vstupní obraz, $B(x, y)$ je výstupní binární obraz.
- Pro každý pixel obrazu vstupního obrazu $G = (x, y)$:

$$B(x,y) = \begin{cases} 0\ (black), & G(x,y) < threshold \\ 1\ (white), & G(x,y) \geq threshold \end{cases}$$



a) b)

Obrázek 3.2: a) Obrázek ve stupních šedi (8b), b) Černobílý obrázek (1b)

3.4 Segmentace kontury

Při segmentaci kontury jsou používány základní znalosti z analytické geometrie ve dvourozměrném prostoru. Pro úplnost jsou stručně uvedeny použité vzorce. Kompletní definice a odvození lze nalézt např. v [6].

Vektor

Směrový vektor **u** je dán dvěma body A a B, přičemž A je počáteční bod a B je koncový bod, pak

$$\mathbf{u} = \overrightarrow{AB} = B - A = (b, -a)\,.$$

Normálový vektor **n** je kolmý ke směrovému vektoru $\mathbf{n} = (a, b)$.

Velikost vektoru

Velikost vektoru **u** je dána druhou odmocninou součtu druhých mocnin složek vektoru

$$|\mathbf{u}| = \sqrt{u_1^2 + u_2^2}.$$

Skalární součin

Skalární součin dvou nenulových vektorů **u**, **v** je reálné číslo, které dostaneme jako součet součinů složek vektorů

$$\mathbf{u} \cdot \mathbf{v} = u_1 \cdot v_1 + u_2 \cdot v_2.$$

Přímka

Přímka p je v obecném tvaru v rovině definována jako

$$am_x + bm_y + c = 0,$$

kde bod $M = [m_x, m_y]$ je libovolný bod přímky, $\mathbf{n} = (a, b)$ je normálový vektor přímky a c je posunutí vůči počátku souřadného systému.

Odchylka přímek

Odchylka ω přímek u a v (obr. 3.3) je dána podílem skalárního součinu a součinu velikostí směrových vektorů

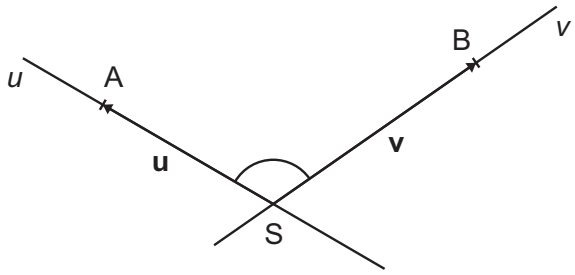
$$cos\omega = \frac{\mathbf{u} \cdot \mathbf{v}}{|\mathbf{u}| \cdot |\mathbf{v}|} = \frac{u_1 \cdot v_1 + u_2 \cdot v_2}{\sqrt{u_1^2 + u_2^2} \cdot \sqrt{v_1^2 + v_2^2}}.$$

Vzdálenost bodu od přímky

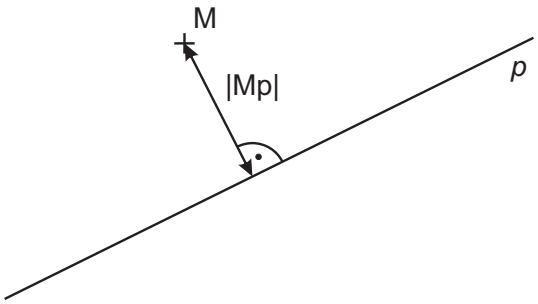
Vzdálenost bodu M od přímky p v rovině (obr. 3.4)

$$|Mp| = \frac{(am_x + bm_y + c)}{\sqrt{a^2 + b^2}},$$

kde $M = [m_x, m_y]$, $\mathbf{n} = (a, b)$ je normálový vektor přímky p a c je posunutí přímky p vůči počátku souřadného systému.



Obrázek 3.3: Odchylka ω přímek u a v



Obrázek 3.4: Vzdálenost bodu M od přímky p

3.5 Korelace

Cílem výpočtu korelace je zjištění podobnosti dvou diskrétních signálů a získání vlastností této podobnosti [8].

Mějme dva diskrétní signály $x[n]$, $y[n]$. Pak jejich korelace je definována

$$\begin{aligned} r_{xy}[m] &= \sum_{n=-\infty}^{\infty} x[n] \cdot y[n-m] \\ &= \sum_{n=-\infty}^{\infty} x[n+m] \cdot y[n], m = 0, \pm 1, \pm 2, \ldots \end{aligned}$$

kde m vyjadřuje časové posunutí a index xy označuje signály, s nimiž je korelace prováděna.

V časovém kroku, kde je korelace největší, je velmi pravděpodobné, že se dané signály shodují.

Kapitola 4

Návrh

Požadavkem pro dobré řešení bylo nalézt postup zpracování testovacích objektů a vyhledání jejich podobnosti na základě tvaru objektu. Při návrhu byl brán ohled na již vytvořené projekty a na požadavky od budoucích uživatelů programu. O získávání testovacích dat se zabývá kapitola 6.

Při návrhu aplikace byly kladeny důrazy na:

- *Open-source* – Program je šířen zkompilovaný i ve formě zdrojových kódů.
- *Multi-plaformnost* – Program je funkční na operačním systému Linux i Windows.
- *Přehlednost výstupního souboru* – Intuitivní vypisování možných kombinací rekonstrukce pro každý objekt.
- *Použitelnost v praxi* – Využití programu archeology nebo paleontology při rekonstrukci objektů ve 2D.

4.1 Zpracování fotografií

Fotografie je vhodné upravit tak, aby objekt zabíral cca 70 % plochy. Musí zůstat zachovány poměry velikostí objektů. Pokud bylo při fotografování špatné světlo, nebo špatně nastavený fotoaparát, musí se upravit jas a kontrast fotografie.

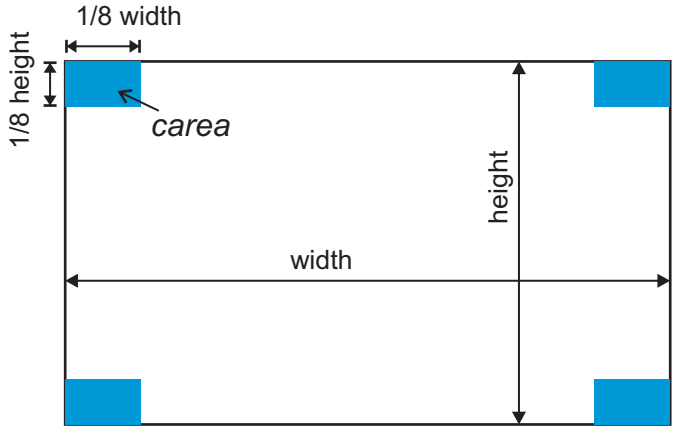
Pro použití v praxi jsou kladeny nároky na usnadnění práce, proto je třeba zpracovávat fotografie hromadně. Barevné fotografie jsou pro zvlolenou metodu rekonstrukce zbytečné, proto byly ihned převedeny na šedotónový obraz (viz. sekce 3.2).

4.2 Zpracování kontur

Aby mohly být nalezeny kontury v obraze, je nutné převést vstupní šedotónový obraz do černobílé reprezentace, čímž získáme lepší kontrast mezi mezi střepem a jeho okolím. Proto je nutné určit správný práh intenzity. Základ této problematiky byl diskutován v sekci 3.3.

Algoritmus vyrovnává hodnotu prahu u hodně světlých a tmavých obrázků pomocí experimentálně zjištěných hodnot *thresholdu*, které jsou náležitě ošetřeny.

Definice 4.1: Nechť *width* a *height* jsou šířka a výška vstupního obrazu $G(x, y)$. Pak oblast rohu obrazu *carea* je definována poměrem 1/8 velikosti *width* na šířku a 1/8 velikosti *height* na výšku (obr. 4.1).



Obrázek 4.1: Oblasti rohů v obraze

Algoritmus 4.2: *Výpočet thresholdu.*

- Pro každou oblast rohu *carea*:
  - vypočítej aritmetický průměr a intenzity každého pixelu v oblasti rohu obrazu
- Přes všechny čtyři *carea* vypočítej aritmetický průměr hodnot, pak urči výsledný *threshold*:

$$threshold = \begin{cases} threshold/0.3, & threshold < 64 \\ threshold, & jinak \\ threshold * 0.6, & threshold > 186 \end{cases}$$

4.2.1 Zvolení největší kontury

Vlivem stínů a odlesků mohou vzniknout jisté nepřesnosti v detekci kontur v obraze. Předpokladem je, že v obraze se vyskytuje pouze jeden objekt k porovnání, pak v obraze bude jedna kontura objektu a kontury šumu. Problém vybrání kontury objektu byl nejdříve řešen pomocí vyhledávání ve stromu kontur, což nebylo optimální řešení. Lepší řešení je vypočítávat obsah všech kontur a vybrat pouze konturu zabírající největší plochu v obraze.

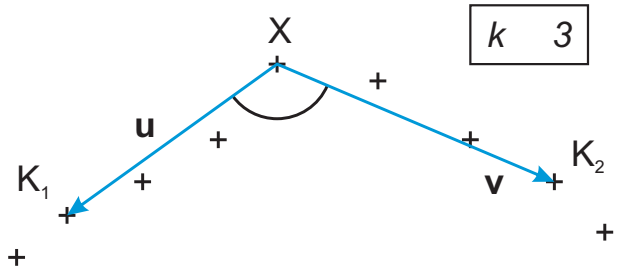
Algoritmus 4.3: *Výběr největší kontury.*

- počáteční hodnota maximálního obsahu $area = 0$
- Pro každou konturu k v seznamu kontur *cont*:
  1. vypočítej obsah S kontury k
  2. jestliže $S > area$ pak:
     - označ konturu jako největší
     - nově nastav hranici $area$ na S

4.2.2 Segmentace kontury

Kontura je reprezentována posloupností bodů jdoucích po sobě ve směru hodinových ručiček. Snadnějšímu vyhledávání napomáhá rozdělení kontury na více částí. Optimální je rozdělení podle skutečných rohů objektu. To ale, vzhledem k možným tvarům objektů, není zcela triviální řešení.

Definice 4.4: Nechť X je libovolný bod kontury a K_1, K_2 jsou sousední body bodu X (pozice těchto bodů v kontuře je $\pm k$, vůči pozici bodu X). Mějme dva vektory $\mathbf{u} = \overrightarrow{XK_1}$ a $\mathbf{v} = \overrightarrow{XK_2}$. Pokud úhel ω svíraný těmito vektory je větší jak 30° a menší jak 150°, je bod X možný roh objektu.



Obrázek 4.2: Detekce rohu v kontuře

4.2.3 Normalizace hran

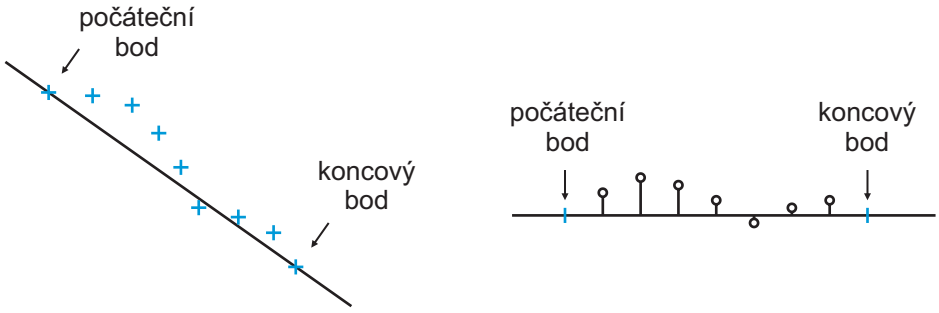
Definice 4.5: Nechť A, B jsou dva sousedící rohy kontury a p je posloupnost bodů v kontuře mezi těmito body. Pak uspořádaná trojice $h = \{A, B, p\}$ nazýváme *hrana*.

Reprezentace hran řeší otázku, jakým způsobem normalizovat a ukládat data. Prosté uložení posloupnosti bodů je zcela nedostačující. Hrana se musí převést do jedné roviny rovnoběžné s osou x. Tím se dosáhne pomocí výpočtu vzdálenosti bodu od přímky (viz. 3.4).

Algoritmus 4.6: *Normalizace hrany*

- Pro každou hranu $h = \{A, B, p\}$ v kontuře:
  1. vytvoř přímku px danou dvěma body A, B
  2. pro každý bod H hrany h:
     - vypočítej vzdálenost v bodu H od přímky px
     - ulož vzdálenost v

Tím převedeme hranu na diskrétní signál (obr. 4.3b), který se bude pomocí korelace porovnávat s ostatními hranami.



Obrázek 4.3: Normalizace detekované hrany objektu

Kapitola 5

Implementace

V této kapitole je podrobněji popsána vlastní implementace programu. Jsou zde popsány použité implementační prostředky. Program se skládá z jednoho hlavního programu *archeo*. Program využívá knihovnu pro zpracování obrazu.

5.1 Implementační prostředky

Pro implementaci byl zvolen programovací jazyk C++, který vznikl v roce 1983 v Bellových laboratořích. Autorem jazyka je Bjarne Stroustrup, který C++ koncipoval jako rozšíření jazyka C. Byly přidány především třídy, šablony, zpracování výjimek a přetěžování operátorů. C++ povoluje různé programátorské přístupy, jako je procedurální programování, objektově orientované programování či generické programování.

Volba implementačního jazyka byla dána znalostí daného jazyka, požadavkem na přenositelnost a kompatibilitu s následně vybranou knihovnou pro zpracování obrazu.

Pro implementaci hromadného zpracování dat byl vybrán jazyk Python. Python je interpretovaný objektově orientovaný programovací jazyk, který v roce 1990 navrhl Guido van Rossum.

5.2 Knihovna OpenCV

Knihovna OpenCV (Open Computer Vision Library) byla původně vyvinuta firmou Intel. Je nezávislá na platformě, a je optimalizována a přizpůsobena pro zpracování obrazových informací. Vyniká bohatou zásobou funkcí zaměřených nejen na zpracování obrazu a videa, ale i tvorbu uživatelského rozhraní. Následuje výčet funkcí a datových typů použitých v programu se stručným popisem. Úplná specifikace je dostupná v manuálu [4].

5.2.1 OpenCV datové typy

- `IplImage*`: Typ ukazatele na strukturu s informacemi o načteném obrázku. Je využíván ve všech funkcích, kde je očekáváno zpracování obrazu, např. vykreslení obrazu,

operace s obrazem. Jsou používány tyto položky struktury:

  - `width` – šířka obrazu v pixelech
  - `height` – výška obrazu v pixelech

- `CvPoint`: Typ struktury pro uložení obrazového bodu. Bod je definován jako dvojice souřadnic $A = [x, y]$.
- `CvMemStorage*`: Ukazatel na strukturu pro alokování paměti.
- `CvSeq*`: Ukazatel na strukturu pro uložení sekvence.
- `CvScalar`: Typ struktury pro uložení informace o obrazovém bodu.

5.2.2 OpenCV funkce

- `IplImage* cvCreateImage(size, depth, channel)`: Vytvoří hlavičku a alokuje data pro obrázek o velikosti `size`, bitové hloubce `depth` a počtu barevných kanálů `channel`. Vrací ukazatel na tato nově alokovaná data.
- `IplImage* cvLoadImage(filename, depth)`: Funkce načte obrázek ze souboru `filename`, naalokuje správnou velikost pro data a vrací ukazatel na ně. Podle hodnoty `depth` se určuje barevnost obrázku.
- `CvMemStorage* cvCreateMemStorage(block=0)`: Vytvoří paměťový prostor o velikosti `block` (0=64KB). Vrací ukazatel na právě vytvořený prostor.
- `char* cvGetSeqElem(seq, index)`: Vrací ukazatel na element sekvence `seq` na indexu `index`.
- `void cvReleaseImage(image)`: Uvolní obrázek `image` z paměti.
- `void cvNamedWindow(name, flag=1)`: Vytvoří a pojmenuje okno s názvem `name`. `Flag` udává, jakým způsobem se okno otevře (1=automatická velikost).
- `void cvShowImage(name, image)`: Zobrazí obrázek `image` do okna se jménem `name`.
- `int cvWaitKey(delay=0)`: Čeká na stisk klávesy. `Delay` určuje maximální dobu čekání v ms (0=neomezeně). Vrací ASCII hodnotu stisknuté klávesy.
- `CvPoint cvPoint(x, y)`: Vrací strukturu `CvPoint` naplněnou parametry `x`, `y`.
- `void cvThreshold(src, dst, thres, val, type)`: Aplikuje převedení barevného obrazu `src` do šedotónového podle hranice `thres`. Hodnota `val` určuje rozsah intenzity a `type` metodu prahování. Výsledný šedotónový obraz uloží do `dst`.
- `int cvFindContours(img, storage, contour, size, mode)`: Vyhledá kontury v obrázku `img`. Uloží je do kontejneru `storage`, kde `contour` ukazuje na první konturu. `Size` je velikost kontury a `mode` je způsob vyhledání kontur.

- `void cvDrawContours(img, contour, excolor, holecolor, max=-1, thick=1, conn=8)`: Vykreslí kontury `contour` do obrázku `img`. `Excolor` je barva, kterou se vykreslí okraj kontury, `holecolor` barvou se vykreslí výplň kontury. Parametr `max` udává, kolik kontur má být vykresleno (-1=všechny uložené), `thick` je šířka vykreslení okraje kontury (1=1px).
- `double cvContourArea(contour)`: Vypočítá a vrací obsah kontury `contour`.

5.3 Python

Vedle základních funkcí jsou použity moduly pro čtení z příkazové řádky `sys` a pro spuštění příkazu jako v příkazové řádce `commands`, kterými se volá přeložený C++ program. Pro zpracování parametrů příkazové řádky jsou použity regulární výrazy (modul `re`).

5.4 Vlastní implementace

5.4.1 Nově definované datové typy

Následuje stručný výčet nově vytvořených datových typů a tříd. Podrobná specifikace a popis metod je uveden v hlavičkových souborech programu.

- `TVectPts`: Vektor bodů `CvPoint` pro uložení kontury a segmentů kontury.
- `TVectChar`: Vektor řetězců pro uložení názvů obrázků ve vstupním adresáři.
- `TVectDbl`: Vektor reálných čísel pro práci s vektory přímky.
- `TVectInt`: Vektor celých čísel pro uložení pozice bodů v kontuře.
- `CImageX`: Třída reprezentující obraz typu IplImg*. Umožňuje načíst obrázek ze souboru, zobrazit ho, převést ho do černobílé reprezentace a vyhledat kontury.
  - `void LoadImage(const char* filename)`: Načte obrázek ve stupních šedi z umístění `filename`.
  - `void Threshold(int flags=THRESHOLD)`: Převede obrázek do černobílé reprezentace. Jako parametr je volba implementovaného thresholdu autorem (`THRESHOLD`) nebo adaptivního thresholdu z OpenCV (`THRESHOLD_ADAPTIVE`).
  - `void ShowImage(const char* windowname`: Zobrazí obrázek s v okně s názvem `windowname`.
- `CVectorX`: Třída reprezentující vektor. Umožňuje provádět skalární součin s dalším vektorem, zjistit délku vektoru a určit úhel svírající s dalším vektorem.
- `CLine2D`: Třída reprezentující přímku ve 2D v obecném tvaru. Vypočítá parametry přímky ze dvou bodů. Umožňuje vypočítat vzdálenost bodu od přímky.

5.5 Rozhraní a ovládání

Výsledný program je konzolová aplikace. Byl více kladen důraz na funkčnost, než na grafické uživatelské rozhraní. Program se spouští zadáním příkazu do příkazové řádky (konzole). Parametry a přepínače uvedené v kulatých závorkách jsou povinné, v hranatých závorkách nepovinné. Při spuštění je možné zadat tyto přepínače a parametry.

synopsis:	`./archeo (-f file|-d directory) [-w] [-o outfile]`
`--help`	Vypíše nápovědu o použití programu.
`-f file`	Načte jeden obrázek `file`.
`-d directory`	Načte obrázky z adresáře `directory`.
`-w`	Pro každý obrázek vygeneruje okno s vyhledanou konturou.
`-o outfile`	Výstup programu bude uložen do souboru `outfile`.

Program podporuje obrázky typu `jpg`, `png` a `gif`. Pokud jsou data v jiném formátu, musí se překonvertovat externím programem.

Standardně se výstup souboru vypisuje přímo do konzole. Pokud s programem bude pracovat pouze jeden člověk a jeho výsledky bude distribuovat, je vhodnější data ukládat do souboru. Parametr `-w` způsobí, že se pro každý obrázek vygeneruje okno s detekovanou konturou objektu a s rohy tohoto objektu.

Kapitola 6

Testování

Testování probíhalo na třech druzích objektů, které se stupňovaly od nejlehčích po nejsložitější. Jako první jsou to nakreslené základní tvary ve vektorovém grafickém programu, exportované do rastrového obrázku. Dalším typem objektů jsou talíře rozbité pro tento účel. Pro lepší testování byl vybrán talíř barvy bílé a černé, bez vzorů. Nejzajímavějšími a nejkomplikovanějšími objekty pro testování jsou archeologické střepy hliněných nádob.

6.1 Testovací data

Testované objekty bylo nutné správně vyfotografovat. Nejdříve bylo vyzkoušeno fotografování na bílém podkladu. Tato metoda se nejevila jako nejlepší, protože i při použití více světelných zdrojů osvětlujících plochu se střepem, se na fotografii okolo střepu objevovaly značné stíny. Samozřejmě tento podklad byl zcela nevhodný pro bílý talíř.

Jako další byl vyzkoušen zelený podklad. Zde byla myšlenka nahrazení všech odstínů zelené barvy za bílou. Toto řešení se zdá zajímavé, ale mohlo by být tématem pro samostatnou bakalářskou práci.

Pro bílý talíř byl vyzkoušen černý textilní podklad, který se jevil jako nejlepší řešení pro tento druh testovacích dat. Nebyly vidět stíny, ani odlesky.

Pro černý talíř a archeologické střepy byla vybrána technika focení na podsvětleném podkladu. Střepy byly umísťovány na sklo podlepené pauzovacím papírem a podsvětlené žárovkou. Tato konstrukce je vidět na obrázku 6.1. Metoda eliminuje stíny a nezávisí tolik ani na barvě střepu.

Aby bylo zachováno měřítko, je nutné fotit střepy vždy ze stejné vzdálenosti, například pomocí trojnožky. I přes toto opatření je vhodné umístit vedle střepu pravítko, ze kterého lze snadno určit rozměr střepu.

6.2 Vyhledání a segmentace kontury

Na následujících obrázcích je možné vidět porovnání fotografie střepu, černobílého obrazu střepu a obrazu s vyhledanou konturou, ve kterém jsou zeleně spojené rohy objektu. Je

Obrázek 6.1: Konstrukce focení střepů na podsvíceném podkladu

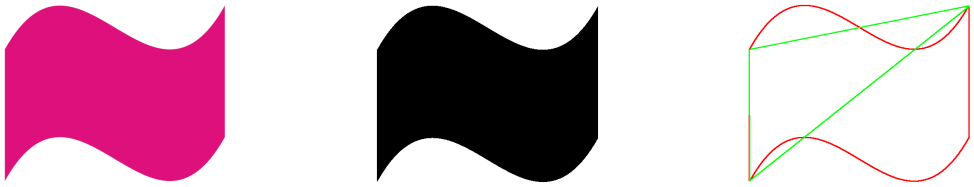
zde uveden pouze zlomek testovacích dat a jejich výsledků. Všechny je možné nalézt na přiloženém CD.

Základní tvary

Na základních tvarech (obrázky 6.2, 6.3, 6.4) se nejdříve testovalo, zda bude správně nalezena kontura jednoduchého obejktu. Pro všechny základní tvary byla správně nalezena kontura a dobře detekované hrany objektu.

Obrázek 6.2: `(tvar-01.png)`: deformované logo FIT.

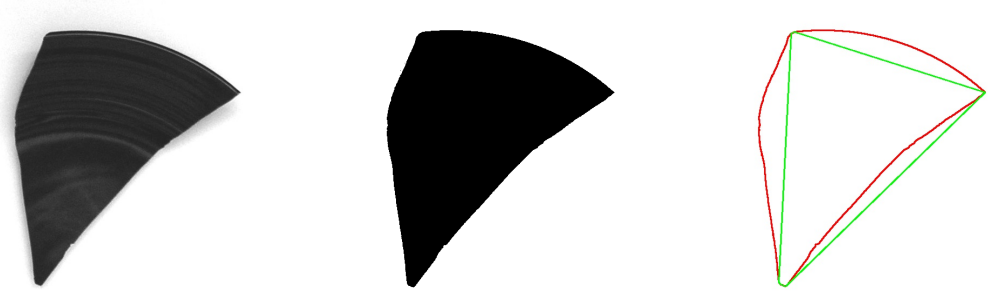
Obrázek 6.3: `(tvar-02.png)`: kříž.



Obrázek 6.4: `(tvar-03.png)`: vlajka.

Střepy talířů

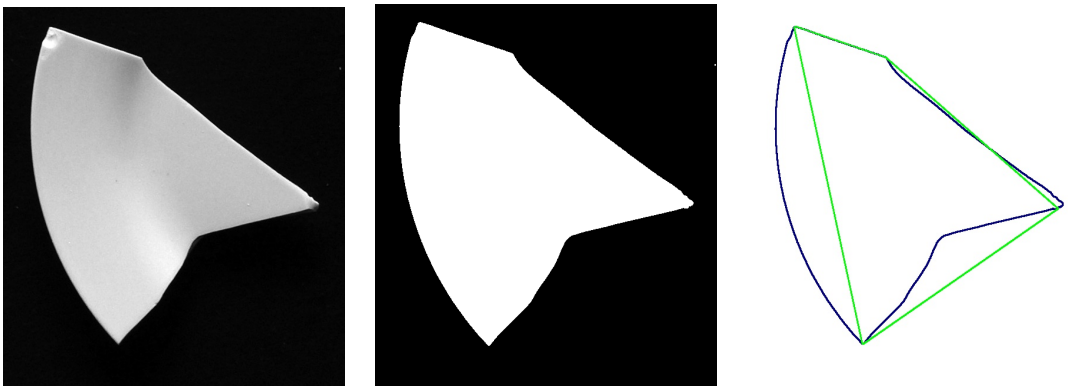
Zde jsou uvedeny výsledky pro střepy z talířů (obrázky 6.5, 6.6). Při vyhledávání kontur vyvstává problém s barevností glazury střepu. Pokud bylo použito inverzní pozadí vůči barvě glazury, tak bylo dosaženo dobrých výsledků.



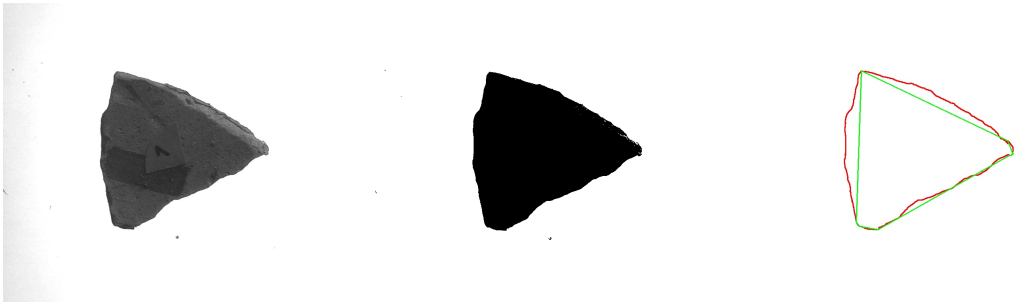
Obrázek 6.5: `(talir-cerny-01.jpg)`: střep z černého talíře.

Archeologické střepy

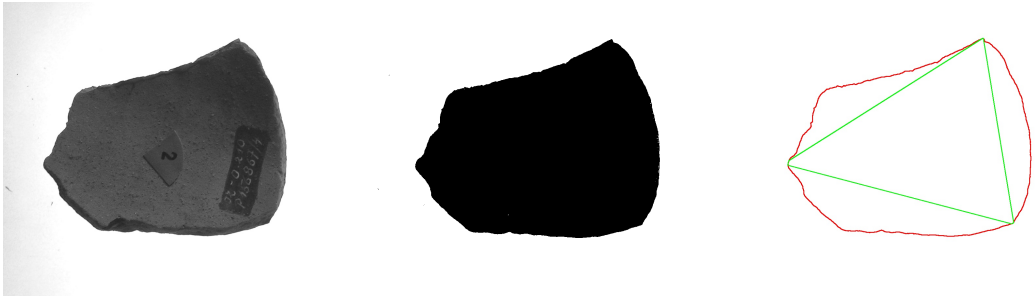
Při testování archeologických střepů (obrázky 6.7, 6.8) musely být experimentálně zjištěny hodnoty úhlů určujích, zda se jedná o roh kontury, či nikoli. Při změně těchto konstat bylo dosaženo uspokojujích výsledků.



Obrázek 6.6: `(talir-bily-05.jpg)`: střep z bílého talíře.



Obrázek 6.7: `(strep-01.jpg)`: střep z hliněné nádoby.

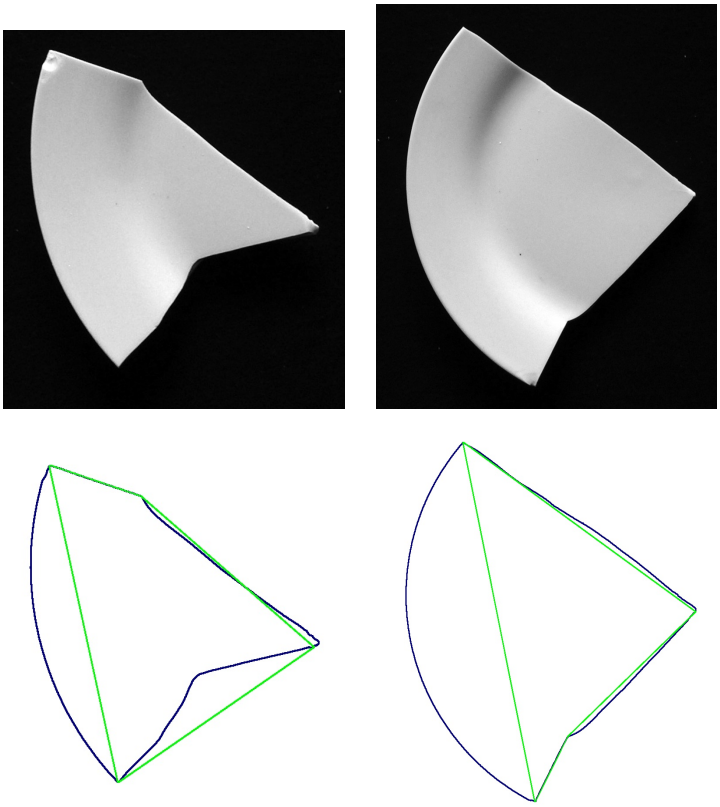


Obrázek 6.8: `(strep-02.jpg)`: střep z hliněné nádoby.

6.3 Párování střepů

Párování střepů z talířů je uspokojivé. Při párování hliněných střepů dochází vlivem opotřebení hran střepu k nepřesnostem.

Nalezené kombinace párů pro střepy z talířů jsou na následujících obrázcích.



Obrázek 6.9: Správně nalezená kombinace střepů z talíře.

Kapitola 7

Závěr

Během vývoje programu jsem se potýkala s celou řadou problémů, počínaje porozuměním datových typů a jejich uložení v knihovně OpenCV a konče odstraňováním vyjímečných situací při běhu programu. Vzhledem k této skutečnosti se mi nepodařilo program rozvinout tak, jak jsem původně chtěla, nicméně jsem zadání práce splnila a program je přichystaný k otestování na Ústavu archeologie Filozofické fakulty Masarykovy univerzity v Brně.

V budoucnosti bych chtěla na práci pokračovat. Bude potřeba více zautomatizovat zpracování fotografií a vylepšení vyhledání párů. Lepší zautomatizování tkví v detekci objektu na fotografii, kde se fotografie ořízne tak, aby objektu byl ve středu obrazu a zabíral cca 70 % plochy obrazu. Dále v automatické opravě jasu a kontrastu. Vyhladávání by mělo být uzpůsobeno tvarům střepů v dané testovací sadě.

Bude vytvořeno grafické prostředí, které bude vykreslovat ke zvolenému střepu ze sady možné kadidáty do páru. Tyto střepy by již měly mít graficky označené hrany, které k sobě pravděpodobně pasují.

Literatura

[1] Flavio BORTOLOZZI. Document reconstruction based on feature matching. In *SIBGRAPI '05: Proceedings of the XVIII Brazilian Symposium on Computer Graphics and Image Processing*, page 163, Washington, DC, USA, 2005. IEEE Computer Society. ISBN 0-7695-2389-7.

[2] Helena Cristina da Gama LEITÃO and Jorge STOLFI. A multi-scale method for the re-assembly of fragmented objects.

[3] Martin P. EVISON. Computerised 3d facial reconstruction.

[4] Intel Corporation, USA. *Open Source Computer Vision Library*, december 2000.

[5] Přemysl KRŠEK. *Základy počítačové grafiky.* Studijní opora, FIT na VUT v Brně, Brno, CZ, 2006.

[6] Jan POLÁK. *Přehled středoškolské matematiky.* Prometheus, Praha, CZ, 2002. ISBN 80-7196-196-5.

[7] Patrick De SMET. High-precision recomposition of fragmented 2-d objects, 2007.

[8] Zdeněk SMÉKAL. *Číslicové zpracování signálů.* Skripta, FEKT na VUT v Brně, Brno, CZ, 2006.

[9] Andrew WILIS, Donald ANDERSON, Thad THOMAS, Thomas BROWN, and J. Lawrence MARSH. 3d reconstruction of highly fragmented bone fractures, 2006.

[10] A. R. WILLIAMS. National geographic Česko. *Tutanchamon odhalen*, 2005(6):80–99. ISSN 1213-9394.